IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of	
Eckhard Kruse et al.	Group Art Unit: 2442
Application No.: 10/562,046	Examiner: Jeffrey L. Nickerson
Filed: April 11, 2006	Appeal No.: _____
For: METHOD AND SYSTEM FOR EVENT TRANSMISSION	

APPEAL BRIEF

Mail Stop APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

This appeal is from the decision of the Primary Examiner dated December 10, 2008 finally rejecting claims 21, 22 and 24-36, which are reproduced as the Claims Appendix of this brief.

☐ A check covering the ☐ $ 270 ☐ $ 540 Government fee is filed herewith.

☒ Charge ☐ $ 270 ☒ $ 540 to Credit Card.

.

The Commissioner is hereby authorized to charge any appropriate fees under 37 C.F.R. §§1.16, 1.17, and 1.21 that may be required by this paper, and to credit any overpayment, to Deposit Account No. 02-4800.

I. Real Party in Interest

The present application is assigned to ABB Research Ltd.. ABB Research Ltd. is the real party in interest, and is the assignee of Application No. 10/562,046.

II. Related Appeals and Interferences

The Appellants' legal representative, or assignee, does not know of any other appeal or interferences which will affect or be directly affected by or have bearing on the Board's decision in the pending appeal.

III. Status of Claims

Claims 1-20 and 23 have been canceled. Claims 21-22 and 24-36 have been finally rejected.

IV. Status of Amendments

All amendments prior to the final rejection have been entered. The Examiner has indicated that the "Substitute Amendment After Final" filed May 11, 2009 would be entered for purposes of Appeal, and entry of that Amendment is hereby requested. (See Advisory Action dated May 27, 2009, Page 2, box "7").

A Substitute Amendment After Final Rejection was filed June 10, 2009 to submit Revised Drawings and additional clarifying arguments, no new claim amendments were proposed. In an Advisory Action dated July 13, 2009, the Examiner indicated that this Substitute Amendment would not be entered. In a subsequent telephone conference with Examiner Nickerson on July 20, 2009, Appellants agreed to submit proposed drawing changes and the Examiner indicated that the drawings would be entered to remove the outstanding drawing objection. Accordingly, a "Submission Of Drawings" is filed herewith, entry of which is requested.

V. Summary Of Claimed Subject Matter

Exemplary embodiments are directed to methods for monitoring and/or controlling technical installations, such as production installations. A client

application can be used as an operator interface to perform installation monitoring and control. Events which occur in the technical installation, such as alarm conditions, are supplied via a data capture unit connected to a server.

Appellant' claim 21 encompasses a method whereby a client application need not request event transmission from the server. Rather, Appellants have disclosed a client application which communicates with a "client event service" whereby events can be "logged" by the client event service and an associated "server event service", such that the events can be transmitted to the client application over a communication link between the server and a client without any active request from the client application. As such the client application can function independently of the server.

Referring to specification page 11, lines 21-24 and to Appellants' exemplary Figure 2, a system is illustrated which can manage and transmit events from a server 2 to a client application 4 by which the client can see event handling and data transmission initiated by the server as though they were occurring in a local environment of the client. For events to be transmitted from the server via the communication link 9 to the client application, the event is logged using the server event service 7 and the client event service 6. Events for which logging has been performed are transmitted from the server, via the client event service 6, to a client application 4 using callback functions and an event identifier. An event identifier is illustrated in Figure 3.

A first logging can prompt an initialization of the client/server system. When an event occurs in the Figure 2 system, it is reported to an installation interface 10. If the event in question has been logged, it is transferred from the installation interface to the server event service 7.

Thus, if there is an event that has been detected by the server event service, it is transmitted via the communication link to the client event service based on the request received from the client event service 6; not from the client application 4 to which it is ultimately transmitted. The client event service transmits received events to the client application 4, where the event is reported. By avoiding a need for active

requests from the client application, the client application need not communicate with the server, and the client application can be independent of the server.

In Appellants' exemplary Figure 2, the client application 4 logs a client callback function 41 in the client event service 6 (see client logging function 61) for an event about which the client application 4 is to be notified. The client event service uses the communication link 9 to log a corresponding server callback function 72 in the server event service 7.

Callback functions can be logged for an event with which a same event name is associated with the client and with the server. The client application 4 can call a client logging function from the client event service and provide it with a name of an event in question and with a pointer to the client callback function which is to be logged. The client logging function can log a unique event identifier, and transmit this event identifier together with the event name to a server logging function 71 of the server event service 7. As such, the event name and unique event identifier can be associated with an event which occurs at the installation interface 10.

The server logging function 71 can log a server callback function 72 with the installation interface 10 by transferring the event name. The server logging function can use a server event table 74 to store a data record which contains the event identifier, and a pointer to the server callback function which is to be logged. The server logging function can use the communication link 9 to report back to the client logging function of the client event service that the logging operation has been performed.

An exemplary system as encompassed by Appellants' claim 27 includes a client 1, and a client event service 6 to make requests for event transmission to a server event service 7 of a server 2 which has a server logging function 71. A server event table 74 is formed as a hash table and holds data records which contain at least one event identifier and a pointer to a server callback function which is to be logged. An event queue 75 can hold entries which describe a respective event, and transmit received events to a client application. Thus, the claim 27 system includes a "client event service" and a "server event service", such that received events can be transmitted to a separate client application.

21. A method for managing and transmitting events from a server via a communication link to at least one client, said method comprising:	E.g., Fig. 2, Server 2; client 1, communication link 9; Spec. p. 11, lines 1-6.
logging possible events in a client event service for the purpose of initializing or updating the client,	Client event service 6
logging possible events in a server event service for the purpose of initializing or updating the server,	Server event service 7
transferring detected events which have been logged from an installation interface to the server event service,	Installation interface 10
sending requests initiated by the client event service regarding the detected events to the server event service,	
transmitting the detected events to the client event service on the basis of a request which has been made to the server event service, and	
transmitting events received by the client event service to a client application,	
wherein the client application logs a client callback function in the client event service for every event about which it is to be notified, and the client event service uses the communication link to log a corresponding server callback function in the server event service, and	Client application 4 Client callback function 41 Server callback function 72
wherein to log the callback functions for an event with which a same event name is associated with the client and with the server, the following steps are performed:	E.g., Specification Page 14, lines 2-34
calling, by the client application, a	Client application 4

client logging function from the client event service and providing said function with a name of an event in question and with a pointer to the client callback function which is to be logged, logging, by the client logging function, a unique event identifier, transmitting the event identifier and the event name via the communication link to a server logging function of the server event service, logging, by the server logging function, a server callback function with the installation interface by transferring the event name, storing, by the server logging function in a server event table, a data record, which contains at least the event identifier and a pointer to the server callback function which is to be logged, reporting, by the server logging function, performance of the logging operation to the client logging function of the client event service via the communication link, and logging, by the client logging function, the client callback function by storing a data record in a client event table, the data record containing at least the event identifier and a pointer to the client callback function which is to be logged.	Client logging function 61 Unique event identifier (Fig. 3); E.g., Spec. p. 14, lls. 8-9 Server logging function 71 Server callback function 72 Server event table 74 (see Fig. 3) Data record 73 Client event table 62 (see Fig. 3)
22. The method as claimed in claim 21, wherein the events to be transmitted are detected by a data capture unit in a technical installation and are reported to the installation interface of the server.	Installation interface 10
24. The method as claimed in claim 21, wherein after a client callback function has been logged for the first time the	Request generator 63

client logging function starts a request generator which then makes requests for event transmission to the server event service.	
25. The method as claimed in claim 24, wherein the request generator of the client event service makes the requests for event transmission to the server event service cyclically.	Specification, page 10, lines 3-11
26. The method as claimed in claim 24, wherein events are transmitted by performing the following steps: detecting, by the installation interface, an event which has occurred and calling the server callback function logged for this event, producing, by the server callback function, an entry describing the event in at least one event queue, reading, by the server event service, the entry produced in the event queue upon the next request from the client event service for event transmission, transmitting the entry via the communication link to the client event service, receiving, by the client event service, the entry, ascertaining and calling the client callback function logged for the event, and executing, by the client callback function, a defined action for the corresponding event in the client application.	 Installation interface 10 Event queue 75 Client event service 6 Client callback function 41
27. A system for managing and transmitting events from a server via a	E.g., Figs. 2, 3; server 2; client 1; communication link 9

communication link to at least one client, said system comprising: at least one client, comprising: at least one client event service, for logging possible events, which uses a communication link to make requests for event transmission to a server event service, a server, comprising: at least one server event service, which has at least one server logging function for logging server callback functions, and for logging possible events, and which uses a communication link to transmit events to the client event service, at least one server event table for holding data records which describe a respective logging operation, which server event table is formed as a hash table and holds data records which contain at least one event identifier and a pointer to a server callback function which is to be logged, at least one event queue for holding entries which describe a respective event, and for transmitting received events to a client application, and at least one installation interface which transfers events which have occurred to the at least one server event service.	client 1 Client event service 6 Server event service 7 Server 2 Server logging function 71 Server callback function 72 Server event table 74 Event identifier (Fig. 3) Pointer Event queue 75 Client application 4 Installation interface 10
28. The method as claimed in claim 21, wherein optionally a tidying function of the server event service is called which deletes the server event table and an event queue if the client event service is no longer communicating with the server event service.	Tidying function 77

29. The system as claimed in claim 27, wherein the installation interface is connected to a data capture unit of a technical installation in order to read in events detected by the data capture unit.	Installation interface 10
30. The system as claimed in claim 27, wherein the server event service has at least one server callback function which can be logged for at least one event and which is called when an event for which it is logged occurs.	Server callback function 72
31. The system as claimed in claim 27, wherein the server event service has, for every client event service with which it communicates via a communication link, a separate client data record which respectively contains at least one server event table and at least one event queue.	Client data record 73 Server event table 74 Event queue 75
32. The system as claimed in claim 31, wherein the server event service has a tidying function which deletes the client data record if the associated client event service is no longer communicating with the server event service.	Tidying function 77
33. The system as claimed in claim 31, wherein the server event table is in the form of a hash table and holds data records which contain at least one event identifier and a pointer to a server callback function which is to be logged.	Server event table 74
34. The system as claimed in claim 27, wherein the client event service has at least one client logging function for logging client callback functions, at least one client event table for holding data records which describe the log, and at least one request generator for making cyclic requests for event transmission.	Client logging function 61 Client event table 62 Request generator 63

35. The system as claimed in claim 34, wherein the client event table is in the form of a hash table and holds data records which contain at least one event identifier and a pointer to a client callback function which is to be logged.	Client event table 62
36. The method as claimed in claim 21, wherein events are transmitted by performing the following steps:	
detecting, by the installation interface, an event which has occurred and calling the server callback function logged for this event,	Installation interface 10; Server callback function 72
producing, by the server callback function, an entry describing the event in at least one event queue,	Event queue 75
reading, by the server event service, the entry produced in the event queue upon the next request from the client event service for event transmission,	Server event service 7 Client event service 6
transmitting the entry via the communication link to the client event service,	
receiving, by the client event service, the entry,	
ascertaining and calling the client callback function logged for this event, and	
executing, by the client callback function, a defined action for the corresponding event in the client application.	Client callback function 41

VI. Grounds of Rejection to be Reviewed on Appeal

The final Office Action sets forth one ground of rejection. For purposes of this appeal, the Board is being asked to review the following ground of rejection as set forth in the final Office Action of December 10, 2008:

A. Claims 21-22 and 24-36 were rejected under 35 U.S.C. 103(a) as being allegedly unpatentable over U.S. Patent No. 6,363,421 (Barker), U.S. Patent No. 6,349,333 (Panikatt) and U.S. Patent Pub. No. 2002/0016867 (Kampe).

VII. Argument:

The Examiner Has Failed To Establish A Prima Facie Case Of Obviousness In Rejecting Claims 21-22 And 24-36 As Being Unpatentable Over U.S. Patent No. 6,363,421 (Barker et al), U.S. Patent No. 6,349,333 (Panikatt et al) And U.S. Patent Pub. No. 2002/0016867 (Kampe et al).

Independent claims 21 and 27 recite patentably distinct features that are not disclosed in the documents relied upon by the Examiner, regardless of whether these documents are considered individually or in the combination asserted by the Examiner. For example, none of the cited documents relied upon by the Examiner discloses or suggests Appellants' claim 21 features of a "client event service" and a "server event service" as claimed, for "transmitting events received by the client event service to a client application".

The foregoing features are neither disclosed nor suggested by the Barker, Panikatt and Kampe documents, viewed individually or in the combination relied upon by the Examiner. In the most recent Advisory Action dated July 13, 2009, the Examiner states in the fourth paragraph on page 2,:

> "As such, the examiner maintains that Barker discloses utilizing a combination of software components (Java applets, various interface architectures, such as COBRA, etc) that do, in fact, perform the claimed functionality."

Contrary to this statement, Barker does not disclose Appellants' claim 21 method, because Barker is not directed to managing and transmitting events from a server to a client application using a "client event service" and a "server event service" as presently claimed. These "services" operate using an "event identifier" so that events which occur on the server side of a server/client system can be transmitted to a client application using a server logging function and a server callback function. As recited in claim 21, "a same event name is associated with the client and with the server". As such, events defined locally by a client application can be monitored or controlled as though they are local to the client application, and the events can be transmitted to the client application based on requests initiated by a separate "client event service" and not by the client application itself.

Barker simply does not disclose or suggest a "client event service" and a "server event service" as presently claimed. The cited sections in Barker (col. 4, ll. 19-55; col. 5, ll. 1 to 23; col. 6, l. 54 to col. 7, l. 67; col. 8, l. 58 to col. 9, l. 8, col. 9, l. 23 to col. 10, l. 67; col. 11, ll. 21-29; col. 15, ll. 21 to 45; col. 21, l. 63 to col. 22, l. 23, col. 17, ll. 25 to 50; col. 25, l. 12 to col. 26, l. 10; col. 33, ll. 42 to 50; and col. 38, l. 50-col. 39, l. 16) do not disclose such features. These citations of the Barker patent disclose that client commands generate HTTP requests to an element management system server. The system server gathers information, dynamically generates a web page, and sends results/outputs to a web browser for display.

Barker discloses Java applets which the Examiner's asserts to be Appellants' claimed "client event service". To the contrary, the Java applets are at best client applications. See Barker at col. 4, lines 27-30. Appellants' claims recite a client application which is separate and distinct from a client event service. The Java applets of Barker can not be properly considered to constitute both the presently claimed "client application" and the "client event service". As noted on page 3, lines 13-21 of the present specification, Appellants' system and method are specifically configured so that a client application need not communicate with a server. Barker does not disclose a client event service as presently claimed.

In addition, Barker does not disclose a "client logging function" of a client event service as presently claimed. Claim 21 also recites "logging of possible events in a server event service for initializing or updating the server". The Examiner

equates Appellants' server event service with "various EMS server subcomponents" and refers to Barker's Fig. 4, which contains a multitude of components, as well as Barker at col. 4, lines 37-55. Barker does not disclose a "server event service" as presently claimed.

Barker is also cited as disclosing at col. 11, lines 21-28 Appellants' claimed sending of requests initiated by a client event service to the server event service. The cited section in Barker states that clients register a filter with an Event Distributor of a server to request delivery (via a callback function) of events matching the filter. Barker's registering of a filter, even under a broad interpretation, does not constitute a "client event service" as presently claimed because it is located in an object server (see col. 11, l. 15). The "events" of Barker are therefore not received by a "client event service" of a client and then transmitted to a client application (see Appellants' claim 21). Rather, the events of Barker are directly transferred from an Event Distributor of a server to a client application. In contrast, Appellants' client application does not communicate with a server, but rather with a client event service which is independent of a server. Because Barker is directed to a system which is fundamentally different from that disclosed by Appellants, attempts to read the Barker system on Appellants claims necessarily fail.

Barker's definition of client callback function in Fig. 6 is cited by the Examiner as corresponding to Appellants' claim 21 feature of a client application which "logs a client callback function in the client event service for every event about which it is to be notified". However, Barker's client callback function is merely passed from the client to the server. Again, because Barker's system is fundamentally different from Appellants' claim 21 method, Barker fails to disclose a client application that logs a client callback function (on the client side) in a client event service as presently claimed for every event about which it is to be notified.

The logging of callback functions for an event with which a same event name is associated with the client and with the server, as presently claimed, is also lacking in the documents relied upon by the Examiner. In contrast to the method and system of Barker, Appellants' client application does not communicate with a server, but rather with a "client event service", and detected events are transmitted to the client event service on the basis of a request made to the server event service. The

same event name is therefore used on the client and server. Barker does not disclose these features, and therefore fails to disclose Appellants' claim 21 method.

The Panikatt and Kampe patents fail to overcome the deficiencies of the Barker patent. For example, Panikatt is cited as allegedly teaching a server callback function, and Kampe is cited as teaching a client event table which contains a client callback pointer. These patents do not disclose the features already discussed, even when these patents are considered in combination with the Barker patent.

As such, Appellants' claim 21 is allowable.

Claim 27 recites features similar to those discussed with respect to claim 21. For example, claim 27 recites a system for managing and transmitting events from a server via a communication link to at least one client. The claim 27 system comprises "at least one client event service, for logging possible events" and "at least one server event service" which transmits events to the "client event service". Claim 27 also recites a "server event table" formed as a hash table which holds data records which contain at least one event identifier and a pointer to a server callback function which is to be logged. At least one event queue is provided "for transmitting received events to a client application".

As discussed with respect to claim 21, Barker's disclosed system is fundamentally different from Appellants claimed invention, such that Barker does not disclose a "client event service" and a separate "server event service" for use in transmitting events to a "client application".

All of the remaining claims depend from claims 21 or 27, and add further distinguishing features which are not disclosed or suggested by the documents relied upon in the final rejection, regardless of whether the documents are considered individually or in the combination suggested by the Examiner. As such, these claims are also allowable.

VIII. Claims Appendix

See attached Claims Appendix for a copy of the claims involved in the appeal.

IX. Evidence Appendix

See attached Evidence Appendix for copies of evidence relied upon by Appellant.

X. Related Proceedings Appendix

See attached Related Proceedings Appendix for copies of decisions identified in Section II, supra.

Respectfully submitted,

BUCHANAN INGERSOLL & ROONEY PC

Date October 13, 2009

By: _____
Patrick C. Keane
Registration No. 32858

P.O. Box 1404
Alexandria, VA 22313-1404
703 836 6620

Table of Contents

VIII. CLAIMS APPENDIX

The Appealed Claims

1-20. (Canceled)

21. A method for managing and transmitting events from a server via a communication link to at least one client, said method comprising:

logging possible events in a client event service for the purpose of initializing or updating the client,

logging possible events in a server event service for the purpose of initializing or updating the server,

transferring detected events which have been logged from an installation interface to the server event service,

sending requests initiated by the client event service regarding the detected events to the server event service,

transmitting the detected events to the client event service on the basis of a request which has been made to the server event service, and

transmitting events received by the client event service to a client application,

wherein the client application logs a client callback function in the client event service for every event about which it is to be notified, and the client event service uses the communication link to log a corresponding server callback function in the server event service, and

wherein to log the callback functions for an event with which a same event name is associated with the client and with the server, the following steps are performed:

calling, by the client application, a client logging function from the client event service and providing said function with a name of an event in question and with a

pointer to the client callback function which is to be logged,

logging, by the client logging function, a unique event identifier,

transmitting the event identifier and the event name via the communication link to a server logging function of the server event service,

logging, by the server logging function, a server callback function with the installation interface by transferring the event name,

storing, by the server logging function in a server event table, a data record, which contains at least the event identifier and a pointer to the server callback function which is to be logged,

reporting, by the server logging function, performance of the logging operation to the client logging function of the client event service via the communication link, and

logging, by the client logging function, the client callback function by storing a data record in a client event table, the data record containing at least the event identifier and a pointer to the client callback function which is to be logged.

22. The method as claimed in claim 21, wherein the events to be transmitted are detected by a data capture unit in a technical installation and are reported to the installation interface of the server.

23. (Canceled)

24. The method as claimed in claim 21, wherein after a client callback function has been logged for the first time the client logging function starts a request generator which then makes requests for event transmission to the server event service.

25. The method as claimed in claim 24, wherein the request generator of the client event service makes the requests for event transmission to the server event service cyclically.

26. The method as claimed in claim 24, wherein events are transmitted by performing the following steps:

detecting, by the installation interface, an event which has occurred and calling the server callback function logged for this event,

producing, by the server callback function, an entry describing the event in at least one event queue,

reading, by the server event service, the entry produced in the event queue upon the next request from the client event service for event transmission,

transmitting the entry via the communication link to the client event service,

receiving, by the client event service, the entry,

ascertaining and calling the client callback function logged for the event, and

executing, by the client callback function, a defined action for the corresponding event in the client application.

27. A system for managing and transmitting events from a server via a communication link to at least one client, said system comprising:

at least one client, comprising:

at least one client event service, for logging possible events, which uses a communication link to make requests for event transmission to a server event service,

a server, comprising:

at least one server event service, which has at least one server logging function for logging server callback functions, and for logging possible events, and which uses a communication link to transmit events to the client event service,

at least one server event table for holding data records which describe a respective logging operation, which server event table is formed as a hash table and holds data records which contain at least one event identifier and a pointer to a server callback function which is to be logged,

at least one event queue for holding entries which describe a respective event, and for transmitting received events to a client application, and

at least one installation interface which transfers events which have occurred to the at least one server event service.

28. The method as claimed in claim 21, wherein optionally a tidying function of the server event service is called which deletes the server event table and an event queue if the client event service is no longer communicating with the server event service.

29. The system as claimed in claim 27, wherein the installation interface is connected to a data capture unit of a technical installation in order to read in events detected by the data capture unit.

30. The system as claimed in claim 27, wherein the server event service has at least one server callback function which can be logged for at least one event and which is called when an event for which it is logged occurs.

31. The system as claimed in claim 27, wherein the server event service has, for every client event service with which it communicates via a communication link, a separate client data record which respectively contains at least one server event table and at least one event queue.

32. The system as claimed in claim 31, wherein the server event service has a tidying function which deletes the client data record if the associated client event service is no longer communicating with the server event service.

33. The system as claimed in claim 31, wherein the server event table is in the form of a hash table and holds data records which contain at least one event identifier and a pointer to a server callback function which is to be logged.

34. The system as claimed in claim 27, wherein the client event service has at least one client logging function for logging client callback functions, at least one client event table for holding data records which describe the log, and at least one request generator for making cyclic requests for event transmission.

35. The system as claimed in claim 34, wherein the client event table is in the form of a hash table and holds data records which contain at least one event identifier and a pointer to a client callback function which is to be logged.

36. The method as claimed in claim 21, wherein events are transmitted by performing the following steps:

detecting, by the installation interface, an event which has occurred and ~~calls~~ calling the server callback function logged for this event,

producing, by the server callback function, an entry describing the event in at least one event queue,

reading, by the server event service, the entry produced in the event queue upon the next request from the client event service for event transmission,

transmitting the entry via the communication link to the client event service,

receiving, by the client event service, the entry,

ascertaining and calling the client callback function logged for the event, and

executing, by the client callback function, a defined action for the corresponding event in the client application.

IX. EVIDENCE APPENDIX

NONE

X. RELATED PROCEEDINGS APPENDIX

NONE